WYDAJNOŚĆ ZŁĄCZEŃ I ZAGNIEŻDŻEŃ DLA SCHEMATÓW ZNORMALIZOWANYCH I ZDENORMALIZOWANYCH

Jan Słaby

1. Cel doświadczenia

Zbadanie wydajność złączeń i zagnieżdżeń porównując systemy zarządzania bazami danych oraz zbadanie wpływu indeksowania na czas wykonywania zapytań.

2. Tabela geochronologiczna

Tablica geochronologiczna to schemat przedstawiający historię Ziemi. Powstaje on na podstawie sekwencji procesów geologicznych, a co za tym idzie, ułożenia warstw skalnych. Poniżej znajduje się tabela opracowana przez Międzynarodową Komisję Stratygraficzną (ICS). Tabela przedstawia klasyfikację czterech jednostek geochronologicznych – eonu, ery, okresu i epoki. Z uwagi na wymiar ostatniej jednostki – piętra zostało ono pominięte w tabeli. Do dalszych rozważań, wzięte pod uwagę zostały dane pokrywające się z Fanerozoikiem, Proteozoik i Archaik zostały ominięte.

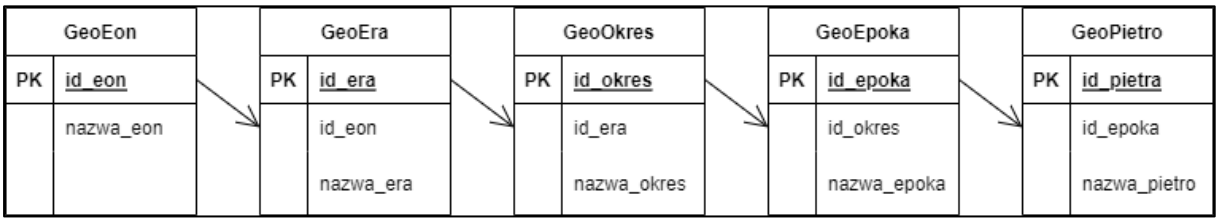
Eon	Era	Okres	Epoka	Datowanie bezwzględne
fanerozoik	kenozoik	czwartorzęd	holocen	11,7 tys.
			plejstocen	2,6 mln
		neogen		23 mln
		paleogen		66 mln
	mezozoik	kreda		145 mln
		jura		201 mln
		trias		252 mln
	paleozoik	perm		299 mln
		karbon		359 mln
		dewon		419 mln
		sylur		443 mln
		ordowik		485 mln
		kambr		541 mln
proteozoik	prekambr*			4,6 mld
archaik				

*Prekambr nie jest ani eonem, ani erą, ani okresem, ani także epoką; jest natomiast bardzo często stosowaną nieformalną jednostką chronologiczną dziejów Ziemi, obejmującą czas od powstania Ziemi do początku okresu kambryjskiego.

Rys. 1 Tabela geochronologiczna

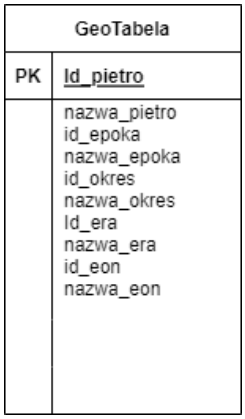
3. Konstrukcja wymiaru geochronologicznego

Pierwszym schematem, który posłuży do pomiarów jest znormalizowana tabela geochronologiczna (Rys. 1), która została podzielona na pięć osobnych części, połączonych kluczem obcym.



Rys. 2 Znormalizowany schemat tabeli geochronologicznej

Drugim przypadkiem będzie „GeoTabela", która będzie postacią zdenormalizowaną pierwszego schematu (Rys. 2). Poszczególne tabele postaci znormalizowanej, zostaną połączone za pomocą złączenia naturalnego. Schemat GeoTabeli przedstawiony został poniżej.



Rys. 3 Zdenormalizowany schemat tabeli geochronologicznej

Żeby stworzyć GeoTabelę (Rys. 3) zostało wywołane zapytanie w następującej fomie:

```
CREATE TABLE GeoTabela AS
(
    SELECT * FROM GeoPietro NATURAL JOIN GeoEpoka
    NATURAL JOIN GeoOkres NATURAL JOIN GeoEra NATURAL JOIN GeoEon
)
```

Rys. 4 Zapytanie tworzące GeoTabele

Utworzenie GeoTabeli (Rys. 3) pozwoliło na szybki dostęp do wszystkich danych tabeli geochronologicznej za pomocą jednego zapytania prostego, nie jest to możliwe w przypadku tabeli znormalizowanej (Rys. 2).

4. Testy wydajności

Testy wykonane na zostały na dwóch popularnych, darmowych rozwiązaniach bazodanowych:

- PostgreSQL
- MySQL

Podczas testów położono nacisk na porównanie wydajności zapytań zagnieżdżonych oraz złączeń, które to zostały wykonane na tabelach o dużej liczbie danych.

4.1 Tabele dodatkowe

Po stworzeniu tabeli zdenormalizowanej i znormalizowanej została stworzona tabela Dziesięć, która zawierała cyfry od 0 do 9 w zapisie dziesiętnym oraz binarnym.

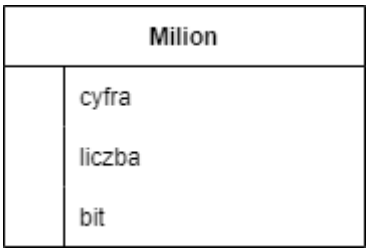
Rys.5 Schemat tabeli Dzisiec

```
CREATE TABLE Dziesiec(
    cyfra INT,
    bit INT
)

INSERT INTO Dziesiec VALUES (0, 00000000),
(1, 00000001),
(2, 00000010),
(3, 00000011),
(4, 00000100),
(5, 00000101),
(6, 00000110),
(7, 00000111),
(8, 00001000),
(9, 00001001)
SELECT * FROM Dziesiec
```

Rys 6. Stworzenie i wypełnienie tabeli Dziesiec

Dzięki tabeli Dziesięć, zostało wykonane autozłączenie, którego skutkiem było wypełnienie tabeli Milion liczbami od 0 do 999 999.



Rys 7. Schemat tabeli milion

```
CREATE TABLE Milion(
    liczba INT,
    cyfra INT,
    bit INT
)

INSERT INTO Milion
SELECT a1.cyfra +10* a2.cyfra +100*a3.cyfra + 1000*a4.cyfra + 10000*a5.cyfra + 10000*a6.cyfra
AS liczba , a1.cyfra
AS cyfra, a1.bit
AS bit
FROM Dziesiec a1, Dziesiec a2, Dziesiec a3, Dziesiec a4, Dziesiec a5, Dziesiec a6;
```

Rys 8. Stworzenie i wypełnienie tabeli milion

5. Konfiguracja sprzętowa i programowa

Testy zostały wykonane na urządzeniu o następujących parametrach:

CPU: Intel Core i5 – 6600 3,30 GHz

RAM: DDR4 16gb (1600MHz)

SDD: WD 1TB

S.O.: Windows 10

MySQL: wersja Community Server 8.0.29

PostgreSQL: wersja 12.3

Testy wykonano wielokrotnie na tym samym urządzeniu.

6. Kryteria testów

W teście zostały wykonane cztery różne zapytania, każde z zapytań zostało powtórzone pięć razy. Sprawdzały one wydajność złączeń i zagnieżdżeń z tabelą geochronologiczną w wersji znormalizowanej (Rys. 2) oraz wersji zdenormalizowanej (Rys. 3).

Aby sprawdzić wpływ indeksowanie na poszczególne tabele, procedura obejmowała dwa etapy:

1. Uruchomienie zapytań bez nałożonych indeksów na kolumny danych
2. Uruchomienie zapytań z nałożonymi na wszystkie kolumny danych, które brały udział w złączeniu

6.1 Zapytania będące przedmiotem testów

1. Zapytanie 1 (1 ZL)

Celem pierwszego zapytania było złączenie tabeli Milion (Rys. 7) ze zdenormalizowaną wersją tabeli geochronologicznej, czyli GeoTabelą (Rys. 3). Do warunku złączenia została dodana operacja modulo, która dopasowywała zakresy wartości złączanych kolumn.

```
SELECT COUNT(*) FROM Milion INNER JOIN GeoTabela ON
(mod(Milion.liczba, 68)=(GeoTabela.id_pietro));
```

Rys 9. Zapytanie 1

2. Zapytanie 2 (2 ZL)

Celem zapytania drugiego było złączenie tabeli Milion ze znormalizowaną wersją tabeli geochronologicznej (Rys. 2)

```
SELECT COUNT(*) FROM Milion INNER JOIN GeoPietro ON
(mod(Milion.liczba,68)=GeoPietro.id_pietro) NATURAL JOIN GeoEpoka NATURAL JOIN
GeoOkres NATURAL JOIN GeoEra NATURAL JOIN GeoEon;
```

Rys 10. Zapytanie 2

3. Zapytanie 3 (3 GZ)

Celem trzeciego zapytania było złączenie tabeli Milion ze zdenormalizowaną wersją tabeli geochronologicznej (Rys. 3). Złączenie zostało wykonane przez skorelowane zagnieżdżenie.

```
SELECT COUNT(*) FROM Milion
WHERE mod(Milion.liczba,68) =
(
    SELECT id_pietro
    FROM GeoTabela
    WHERE mod(Milion.liczba,68)=
    (
        id_pietro
    )
);
```

Rys 11. Zapytanie 3

4. Zapytanie 4 (4 GZ)

Celem czwartego zapytania było złączenie tabeli Milion ze znormalizowaną wersją tabeli geochronologicznej (Rys. 2). Zapytanie wewnętrzne jest złączeniem tabel poszczególnych jednostek geochronologicznych. Złączenie jest tutaj wykonywane przez skorelowane zagnieżdżenie.

```
SELECT COUNT(*)
FROM Milion
WHERE mod(Milion.liczba,68) IN (SELECT GeoPietro.id_pietro
FROM GeoPietro
NATURAL JOIN GeoEpoka
NATURAL JOIN GeoOkres
NATURAL JOIN GeoEra
NATURAL JOIN GeoEon);
```

Rys 12. Zapytanie 4

5. Wyniki testów

Każdy z testów przeprowadzono pięć razy dla wszystkich zapytań. Wyniki testów zostały umieszczone w tabeli poniżej. Wyniki wpisane są w milisekundach.

PostGreSQL bez indeksów:					
	PROBA 1	PROBA 2	PROBA 3	PROBA 4	PROBA 5
1 ZL	182	184	156	204	167
2 ZL	237	237	301	373	220
3 GZ	12674	12205	12183	12220	12862
4 GZ	196	173	186	195	207

PostGreSQL z indeksami:					
	PROBA 1	PROBA 2	PROBA 3	PROBA 4	PROBA 5
1 ZL	162	165	167	141	173
2 ZL	229	230	244	237	217
3 GZ	12271	12773	12769	12654	12169
4 GZ	184	172	226	168	186

MySQL bez indeksów:					
	PROBA 1	PROBA 2	PROBA 3	PROBA 4	PROBA 5
1 ZL	1992	1719	1688	1766	1765
2 ZL	20390	21219	20360	20625	20359
3 GZ	2532	2516	2515	2594	2594
4 GZ	22078	21015	20922	21907	20719

MySQL z ideksami:					
	PROBA 1	PROBA 2	PROBA 3	PROBA 4	PROBA 5
1 ZL	1702	1766	1719	1781	1781
2 ZL	15609	15562	15781	15969	15891
3 GZ	2500	2594	2594	2546	2453
4 GZ	15937	15860	15843	15860	16063

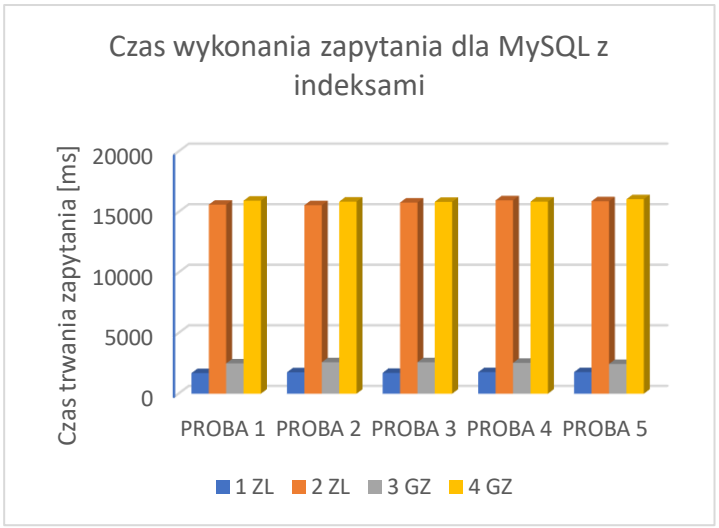
Tab 1. Wyniki pomiarów dla zapytań

Kolejna tabela przedstawia średnie wyniki oraz wartości minimalne. Wyniki są podane w milisekundach.

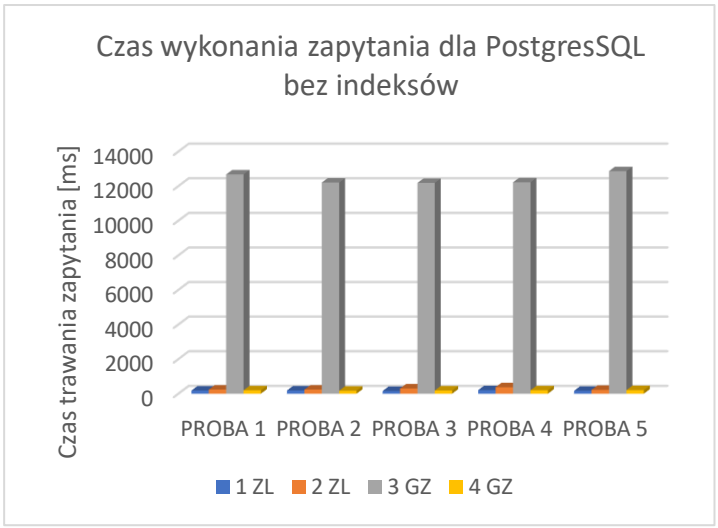
	1 ZL		2 ZL		3 GZ		4 GZ	
	MIN	ŚR	MIN	ŚR	MIN	ŚR	MIN	ŚR
Bez indeksów								
MySQL	1688	1786	20359	20590,6	2515	2550,2	20719	21328,2
PostgreSQL	156	178,6	220	273,6	12183	12428,8	173	191,4
Z indeksami								
MySQL	1702	1749,8	15562	15762,4	2453	2537,4	15843	15912,6
PostgreSQL	141	161,6	217	231,4	12169	12527,2	168	187,2

Tab 2. Wartości minimalne i średnie dla wykonanych pomiarów

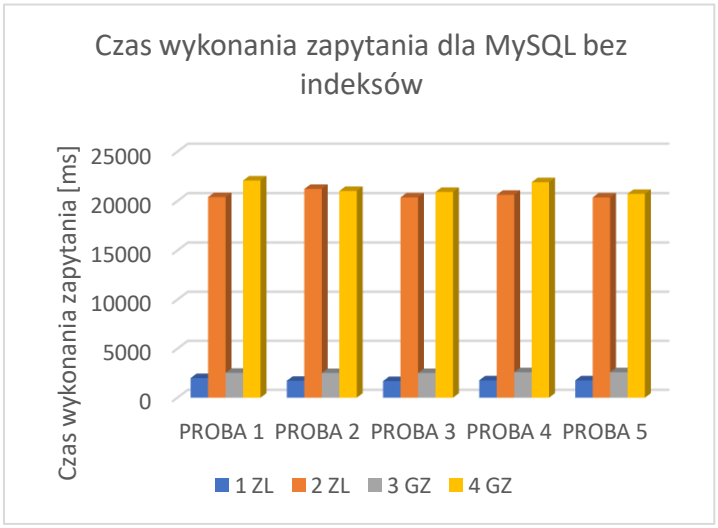
Analizę wyników ułatwią poniższe wykresy słupkowe, przedstawiające czas wykonywania zapytań w każdym rozwiązaniu bazodanowym, z oraz bez indeksów.



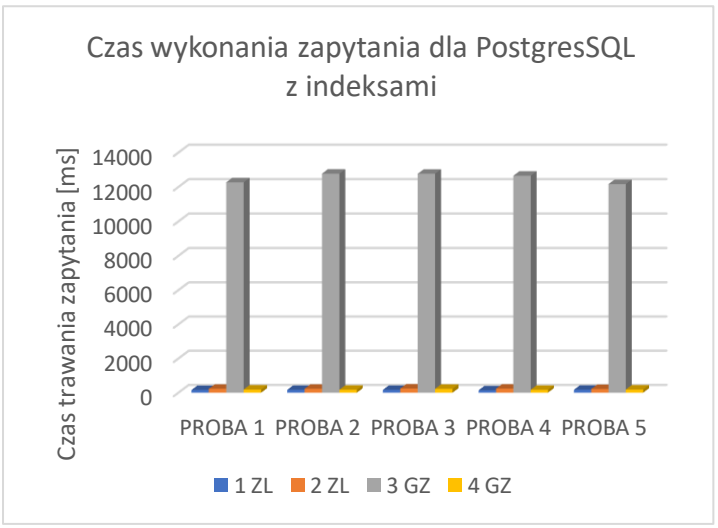
Tab. 3 Czas wykonania zapytania dla MySQL z indeksami



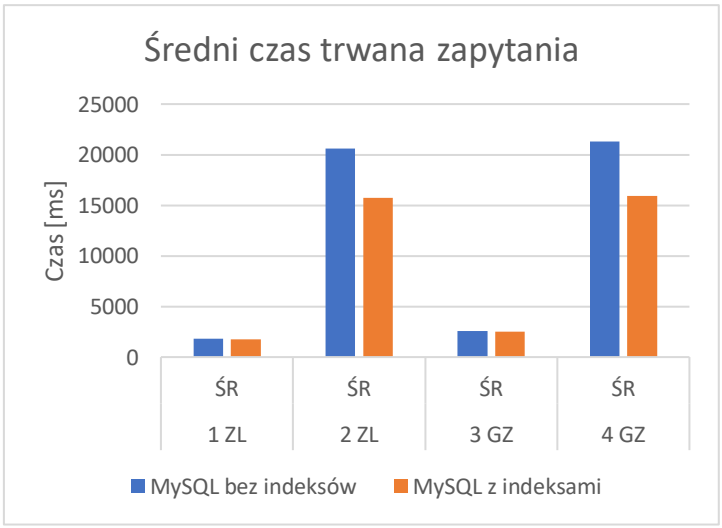
Tab. 4 Czas zapytania dla PosgreSQL z indeksami



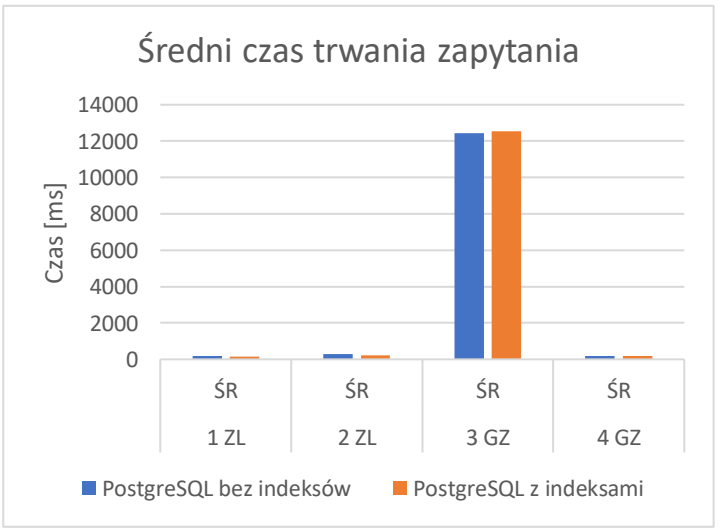
Tab. 5 Czas wykonania zapytania dla PostgreSQL z indeksami



Tab. 6 Czas wykonania zapytania dla MySQL bez indeksów



Tab. 7 Średni czas trwania zapytania dla MySQL z indeksami oraz bez



Tab. 8 Średni czas trwania zapytania dla PostgreSQL z indeksami oraz bez

6. Wnioski

Po dokonaniu analizy danych oraz wykresów, możemy sformułować następujące wnioski:

1. Postać zdenormalizowana jest na ogół szybsza, widać to na przykładzie wyników (Tab. 2), jedynym wyjątkiem jest porównanie zapytania trzeciego i czwartego w systemie PostgreSQL, gdzie zasada jest odwrotna i to postać znormalizowana ma znaczącą przewagę w czasie wykonania.

2. Dodanie indeksów w MySQL znacząco przyśpiesza wykonywanie się zapytań, różnica jest szczególnie widoczna, gdy zapytanie wykonuje się długo i obsługuje postać znormalizowaną, wtedy czas może się skrócić o ¼.

3. Dodanie indeksów w PostgreSQL nieznacznie przyspiesza wykonywanie się zapytań, których średni czas jest znacząco krótszy niż w MySQL, jedynym wyjątkiem jest zapytanie trzecie (GZ 3), gdzie zapytanie obsługujące tabele bez indeksów okazało się szybsze.

4. PostgreSQL jest systemem bardziej wydajnym niż MySQL, czasy wykonywania zapytań potrafią być nawet 10 razy niższe, pomimo, że składnia poleceń obu systemów jest ta sama.

5. Oprogramowanie PostgreSQL daje wyniki zapytań o bardzo zbliżonym czasie.

6. Najwolniej wykonywanym zapytaniem było zapytanie trzecie, którego średni czas w MySQL wyniósł ponad 25 sekund bez zastosowanych indeksów.

7. Wzrost wydajności w przypadku MySQL był o wiele bardziej liniowy i przewidywalny, gdy porównamy poszczególne wersje zapytań i postaci.

7. Podsumowanie

Przeprowadzony eksperyment przyniósł wiele ciekawych, niewidocznych na pierwszy rzut oka wniosków. Mianowicie, nakładanie indeksów usprawnia wykonywanie zapytań, ale zależy to od oprogramowania, na którym pracujemy, tak więc w przypadku PostgreSQL różnice będą znacznie mniejsze oraz mniej odczuwalne, niż w przypadku MySQL, w którym to, podczas pracy, jak wykazuje powyższy eksperyment, naprawdę warto je stosować. Co więcej, przechowywanie danych schemacie zdenormalizowanym usprawnia proces filtracji danych, co nie jest rzeczą oczywistą, bo mogłoby się wydawać, że z powodu lepszej organizacji danych, postać znormalizowana, będzie miała przewagę, ta jedynie jest widoczna, tylko i wyłącznie, gdy oceniamy czytelność schematów, możliwe więc, że postać znormalizowana jest wolniejsza, ale ogólny czas pracy nad bazą danych może zostać być taki sam, gdy weźmiemy pod uwagę czynnik ludzki.

8. Bibliografia

1. Mgr. Inż. Łukasz Jajeśnica, prof.. nadzw. dr hab. Inż. Adam Piórkowski – Wydajność Złączeń i Zagnieżdżeń dla Schematów Znormalizowanych i Zdenormalizowanych
2. Dr. inż. Michał Lupa – Bazy Danych 2022
3. Draw.io
4. https://zpe.gov.pl/a/jak-poznano-dzieje-ziemi/DzTZFjFgV